Postgres 14 и высокие нагрузки

Иван Панченко







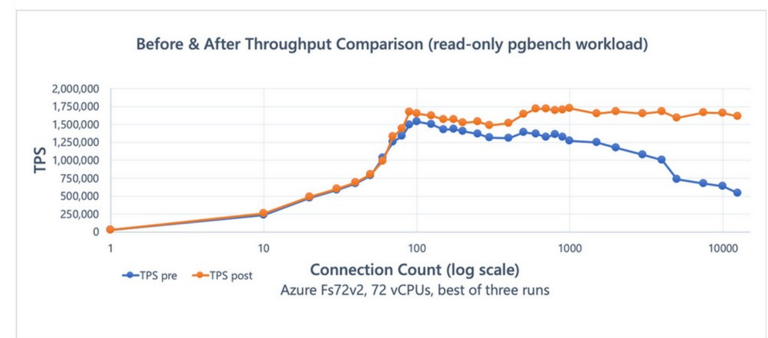
Кратко о Postgres 13

- Сжатие (дедупликация) B-Tree.
- Инкрементальная сортировка: sorted(k1,k2) → sorted(k1,k2,k3)
- PL/PgSQL не ходит в планировщик за простыми операторами
- Вычисление immutable функций на этапе планирования.
- Ускорение **`TRUNCATE`** (один скан shared_buffers вместо трех).
- Частичная декомпрессия TOAST.
- Вакуум индексов параллельно.
- Автовакуум при вставке (visibility map для append-only таблиц)





Повышение масштабируемости по коннектам



Облегчение GetSnapshotData()

Andres Freund:

Analyzing the Limits of Connection Scalability in Postgres

Improving Postgres Connection Scalability: Snapshots




Восходящее удаление

(Bottom-up deletion)

Прежде чем делать split индексной страницы B-Tree при UPDATE, не меняющем ключ индекса, попытаться зачистить старые версии на ней.

Важная оптимизация при большом количестве UPDATE.





Асинхронный APPEND

Секционирование + FDW = Шардинг?

```
SELECT * FROM partitioned_table;


Append
    ->  Async Foreign Scan on partition1
    ........
```

– Непонятно, когда планировщик изберёт такой scan.





Покрывающие индексы SP-GiST

```
CREATE TABLE t (p point, z text);
CREATE INDEX i USING spgist ON (p)INCLUDE (z);
…
EXPLAIN SELECT * FROM t
   WHERE p <@ box(point(5,5),point(6,6));

Index Only Scan using i on t
   Index Cond: (p <@ '(6,6),(5,5)'::box)
```






VACUUM

- Параметры по умолчанию:
  `vacuum_cost_page_miss = 10` → **2** #диски стали быстрее
- Умеет выполняться параллельно с **`CREATE INDEX`** | **`REINDEX CONCURRENTLY`**
- Умеет не заходить в индексы, если сканирование таблицы нашло мало bloat'a.
- **`VACUUM`** `(`**`PROCESS_TOAST`** `OFF)`
- Воркер autovacuum падает вслед за постмастером.





Реплики и репликация (1)

Изменилась реакция реплики на изменение критических параметров

- `max_connections`
- `max_prepared_transactions`
- `max_locks_per_transaction`
- `max_wal_senders`
- `max_worker_processes`

Вместо остановки всей реплики теперь — остановка репликации с предупреждением.





Реплики и репликация (2)

- Изменение `restore_command` не требует перезапуска.
- Для `pg_rewind` можно использовать реплику в качестве source.
- Логическая репликация:
  Передача больших транзакций постепенно.
  `CREATE PUBLICATION … WITH (streaming = on)`
- Логическая репликация в двоичном формате (так быстрее)
  `CREATE|ALTER SUBSCRIPTION … WITH (binary = on)`





pg_stat_*

```
SET compute_query_id = 'on';
```

Заполняется поле `pg_stat_activity.query_id`
чем же, чем `pg_stat_statement.queryid`

- Новое представление:

```
pg_stat_statements_info (
    dealloc bigint, -- сколько раз переполнялось
    stats_reset timestamptz -- когда сбрасывали
)
```




pg_stat_statements

- Различие верхнеуровневых и вложенных запросов:

```
SET pg_stat_statements.track = 'all'
SELECT toplevel, query
  FROM pg_stat_statements
```

- Подсчет обработанных строк в поле `rows` для **CREATE TABLE AS, SELECT INTO, CREATE MATERIALIZED VIEW, FETCH** (раньше не считалось)





Новое представление pg_stat_wal

```
SET track_wal_io_timing = 'on';
SELECT * FROM pg_stat_wal;
wal_records      | 21049276
wal_fpi          | 79     full page images
wal_bytes        | 1242346187
wal_buffers_full | 129358 раз
wal_write        | 129555 раз
wal_sync         | 239    раз
wal_write_time   | 337.535
wal_sync_time    | 409.48
stats_reset      | 2021-05-14 17:04:36.006022+03
```

Есть в pg_stat_statements (wal_records, wal_fpi, wal_bytes)

```
SELECT pg_stat_reset_shared('wal')
```

ResultCache

Новый узел плана запроса. Кеширует результат внутреннего цикла в Nested Loop.

```
SET enable_resultcache = ON;
EXPLAIN (costs off)
    SELECT name FROM customer
    JOIN issue ON issue.customer = customer.id;
```

```
Nested Loop
   ->  Seq Scan on issue
   ->  Result Cache
         Cache Key: issue.customer
         ->  Index Scan using customer_id_idx on customer
               Index Cond: (id = issue.customer)
```

Перестройка индекса в другой tablespace

```
REINDEX (TABLESPACE space_to_move)
    TABLE CONCURRENTLY tablename;
```

Новый индекс создаётся в указанном tablespace.

Нужно, когда кончается место. Альтернативы:

ALTER INDEX … SET TABLESPACE …

ALTER TABLE … SET TABLESPACE …



Расширение `pg_squeeze` CYBERTEC





Конвейерный (pipeline) режим в libpq

PQenterPipelineMode(...)

PQsendQuery(...)

PQsendQuery(...)

PQgetResult(...)

PQgetResult(...)

+ помогает, если:

Много коротких команд, большие задержки в сети.

–

Ждём поддержки в библиотеках более высокого уровня.



FDW

- Новые параметры postgres_fdw:

```
CREATE SERVER … FOREIGN DATA WRAPPER postgres_fdw
    OPTIONS (…  keep_connections 'off',
                batch_size '100');
```

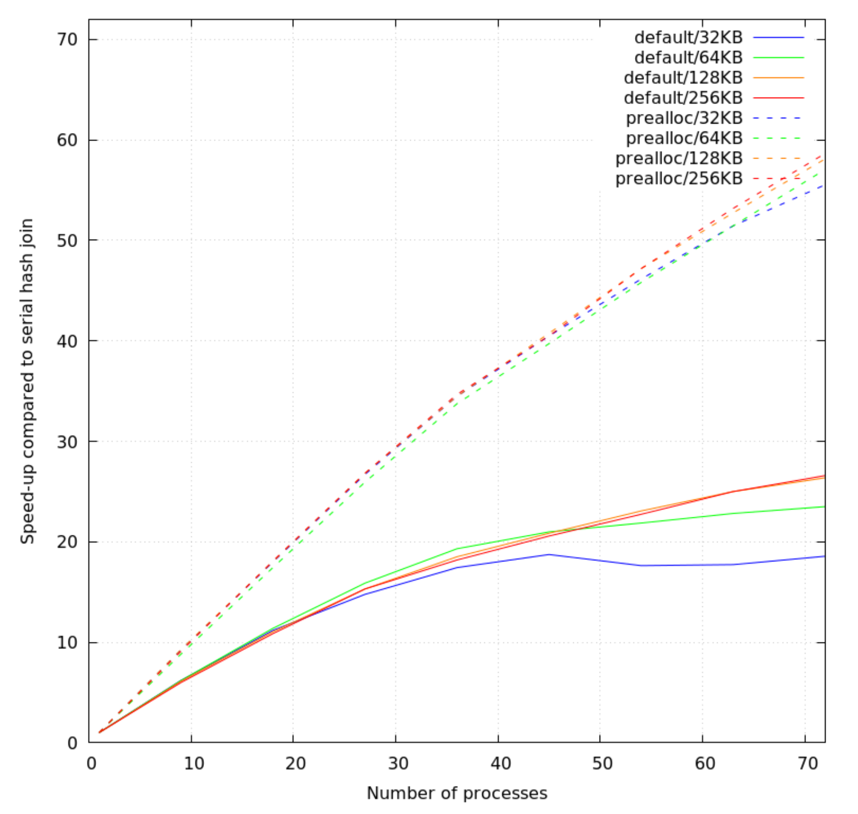
- Автоматический реконнект к удаленному серверу (не в транзакции)





Преаллокация shm для параллельного исполнения запросов

- GUC `min_dynamic_shared_memory`







Статистика

- Статистика по выражениям на таблицах:

```
CREATE STATISTICS s
    ON some_func(r.field1, const1) FROM table r;
```

- Расширенная статистика лучше используется в OR-ах
- В пустой таблице 0 записей





CONCURRENTLY…

- **ALTER TABLE … DETACH PARTITION … CONCURRENTLY;**

  старые транзакции завершаются, видя эту секцию;
  новые уже не видят её, но работают.

- Можно одновременно выполнять
  CREATE INDEX CONCURRENTLY или **REINDEX CONCURRENTLY** по нескольким различным таблицам.





Разное (1)

- Выбор метода компрессии для TOAST (pglz , lz4)

```
./configure —with-lz4
SET default_toast_compression = 'lz4';
ALTER TABLE tab ALTER data SET COMPRESSION lz4;
```

- **`TRUNCATE`** удалённую секцию таблицы (т. е. удаленную таблицу)
- `SET idle_session_timeout = '1000ms';`
- Ускорение зачистки shared buffers при **`TRUNCATE, DROP TABLE`** и т. п
  стараемся обходиться без полного перебора
- `now()` кешируется в рамках транзакции.





Разное (2)

- Использование атрибута компилятора `cold` в ветках с обработкой ошибок.
- Инкрементальная сортировка применяется для оконных функций
- Ускорение построения GiST индекса с помощью пространственного упорядочивания.
- REINDEX применим к секционированным таблицам.
- btree_gist стал PARALLEL_SAFE
- Parallel sequence scan работает пачками блоков, а не по одному.



Что читать

На Хабре:

И. Лёвшин

- Много ли нового в Чёртовой Дюжине?

П. Лузанов

- PostgreSQL 14: Часть 1 или «июльский разогрев» (Коммитфест 2020-07)
- PostgreSQL 14: Часть 2 или «в тени тринадцатой» (Коммитфест 2020-09)
- PostgreSQL 14: Часть 3 или «ноябрьское затишье» (Коммитфест 2020-11)
- PostgreSQL 14: Часть 4 или «январское наступление» (Коммитфест 2021-01)
- PostgreSQL 14: Часть 5 или «весенние заморозки» (Коммитфест 2021-03)





Спасибо за внимание!

Ask Postgres Pro!

https://postgrespro.ru/

PGConf.Russia
20 сентября 2021 г
Следите за анонсами



